10/790509

B. Whether or not independent and dependent claims 1-23 are unpatentable under 35 USC § 103(a) over Arimilli (U.S. Patent No. 6,907,494) (Method and System of Managing Virtualized Physical Memory in a Memory Controller and Processor System) in view of Browning, et al. (U.S. Patent No. 6,918,023) (Method, System, and Computer Program Product for Invalidating Pretranslations for Dynamic Memory Removal).

VII. ARGUMENT

A. Arguments against the rejection under 35 USC § 101 because the claim is allegedly directed to non-statutory subject matter.

1. Arguments regarding independent claim 23.

a. For independent claim 23, the subject matter thereof is not directed to non-statutory subject matter.

Appellant's independent claim 23 recites, "A computer readable medium having computer readable instructions stored thereon for execution by a device to perform a method." Appellant respectfully submits that claim 23 recites tangible elements and instructions that are executed to produce concrete, useful and tangible results.

The Examiner states on page 11 of the June 19, 2008, Final Office Action:

> It is noted that the specification clearly defines "a computer readable medium" include [sic] any medium that can store or transfer information, such as any signal that can propogate over a transmission medium (page 7 lines 8-21). Thus, the claimed subject matter can be interpreted as signal or carrier wave, thereby, the claimed subject matter as recited in claim 23 is non-statutory.

By so stating, the Examiner appears to argue that "a computer readable medium," as recited in Appellant's claim 23, is non-statutory because Appellant's specification allegedly defines "a computer readable medium" as including any signal that can propagate over a transmission medium.

However, page 7, lines 12-21, of Appellant's specification, as originally filed, state:

> A computer readable medium may include any medium that can store or transfer information. Examples of the computer readable medium include an electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable ROM (EROM), a floppy diskette, a compact disk CD-ROM, an optical disk, a hard disk, a fiber optic medium, a radio frequency (RF) link, etc. The computer data signal may include any signal that can propagate over a transmission medium such as electronic network channels, optical fibers, air, electromagnetic, RF links, etc. The code segments may be downloaded via computer networks such as the Internet, Intranet, etc.

That is, Appellant's specification, as originally filed, states that a computer readable medium may include any medium that can store or transfer information, and lists examples of tangible embodiments of the computer readable medium. Hence, Appellant's specification does not define a computer readable medium as including any signal that can propagate over a transmission medium, as stated by the Examiner. Rather, Appellant's specification states that a computer data signal may include any signal that can propagate over a transmission medium.

Moreover, Appellant's independent claim 23 does not recite a computer data signal. Rather, Appellant's claim 23 recites, "A computer readable medium having computer readable instructions stored thereon for execution by a device to perform a method." Hence, the subject matter recited in independent claim 23, e.g., a "computer readable medium having computer readable instructions stored thereon," does not include any signal that can propagate over a transmission medium.

As such, Appellant respectfully submits that independent claim 23 recites statutory subject matter. Accordingly, Appellant respectfully requests reconsideration and withdrawal of the § 101 rejection of independent claim 23.

B. Arguments against the rejections under § 103(a) over the Arimilli '494 reference in view of the Browning '023 reference.

1. Arguments regarding independent claims 1, 8, 19, and 22-23.

a. For independent claims 1, 8, 19, and 22-23, the cited references do not teach, suggest, or render obvious each and every element and limitation.

Appellant does not admit that either the Arimilli '494 reference or the Browning '023 reference are indeed prior art, and reserves the right to swear behind at a future date. Nonetheless, Appellant respectfully submits that the elements and limitations of the claims of the present application, as recited herein, are patentably distinguishable from the teachings of the cited references, either independently or in combination, for at least the following reasons.

With regard to independent claims 1, 8, 19, and 22-23, the Examiner states on pages 3-4, 6, and 10 of the June 19, 2008, Final Office Action:

> Arimilli differs from the claimed invention in not specifically teaching to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use subsequent to when the physical address space is released, wherein registering is triggered by detection that the physical address space that was being used by processes associated with the device has been released; and wherein the registering occurs as the physical address space is released and before release of the virtual address space by the process.

However, the Examiner cites the Browning '023 reference as teaching the above quoted language on pages 4, 6-7, and 10 of the Final Office Action.

Appellant respectfully submits, however, that column 8, line 45, through column 9, line 10, of the Browning '023 reference states:

> FIG. 9 depicts a high level flow chart which illustrates a kernel removing real pages of memory in accordance with the present invention. The process starts as depicted by block 900 and thereafter passes to block 902 which illustrates the kernel starting a memory remove operation. Next, block 904 depicts the kernel atomically incrementing the system memory generation count. The kernel then sets the memory remove in progress flag.
>
> The process then passes to block 906 which illustrates the kernel sending an interprocessor interrupt to all CPUs. Next, block 908 depicts a determination of whether or not the kernel received an acknowledgment of the interrupt from all of the CPUs. If a determination is made that the kernel did not receive an

> acknowledgment of the interrupt from all of the CPUs, the process passes back to block 908. If a determination is made that the kernel did receive an acknowledgment of the interrupt from all of the CPUs, the process passes to block 910 which illustrates the kernel scanning all registered RPN lists and invalidating all entries that correspond to real pages that are within the range of memory to be removed.
>
> Block 912 then, depicts the kernel sending an interprocessor interrupt to all CPUs. Next, block 914 illustrates a determination of whether or not the kernel has received an acknowledgment of the interrupt from all CPUs. If a determination is made that the kernel has not received an acknowledgment from all CPUs, the process passes back to block 914. If a determination is made that the kernel has received an acknowledgment from all CPUs, the process passes to block 916 which depicts the kernel performing memory migration and removal of real pages of memory. Thereafter, block 918 illustrates the kernel clearing the memory remove in progress flag. The process then terminates as depicted by block 920.

By so stating, the Browning '023 reference appears to teach a process in which a kernel first sends and receives acknowledgment of an interprocessor interrupt to and from all CPUs, then scans all registered RPN lists and invalidates all entries that correspond to real pages that are within the range of memory to be removed, and finally performs memory migration and removal of real pages of memory. That is, the kernel receives an acknowledgment of the interrupt and invalidates the relevant pretranslation lists before memory migration and removal, e.g., release, of the real, e.g., physical, pages of memory.

In contrast, Appellant's independent claim 1 recites:

register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use;

wherein registering is triggered by detection that the physical address space that was being used by processes associated with the device has been released; and

wherein the registering occurs as the physical address space is released and before release of the virtual address space by the process.

Appellant's independent claim 8 recites:

register by providing an indication in the virtual memory data structure for the process that the virtual address space is no longer available to the process;

wherein to register is triggered by detection that the physical address space that was being used by processes associated with the device has been released; and

wherein to register occurs as the physical address space is released and before release of the virtual address space by the process.

Appellant's independent claim 19 recites:

removing the object from physical memory when the device is logically disconnected from the computing device; and

providing an indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process as triggered by detection of a physical address space used by the process being released and when the object is removed from physical memory, without removing the representation of the object from the virtual memory data structure for the process.

Appellant's independent claim 22 recites:

> releasing a physical address space when the device has a logical connection removed from the computing device; and
>
> at the release of the physical address space used by the process and before the process has released the virtual address space, registering an indication in a virtual memory data structure for the process that the virtual address space is not available to the process in a manner which does not violate semantics of an operating system.

Additionally, Appellant's independent claim 23 recites:

> releasing a physical address space when the device is logically disconnected from the computing device; and
>
> at the release of the physical address space used by the process and before the process has released the virtual address space, registering an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process in a manner which does not violate semantics for an operating system the computing device.

Further, Appellant respectfully submits that independent claims 1, 8, 19, and 22-23 address, at least in part, memory management for removable memory mappable devices, e.g., devices that can be disconnected from a computing device. In contrast, the Browning '023 reference does not appear to address removable memory mappable devices.

Rather, the Browning '023 reference appears to teach "invalidating specified pretranslations of virtual to physical addresses," and "synchronizing the invalidation process with the memory remove process." (Col. 3, lines 6-8; Col. 3, lines 22-26).

Column 8, line 45, through column 9, line 10, of the Browning '023 reference appear to teach that synchronization of the invalidation process and the memory remove process is achieved by invalidating the specified pretranslations before releasing the real, e.g., physical, pages of memory, as discussed above. That is, the virtual address space is released before the physical address space is released.

However, such a process is not usable in the context of removable memory mappable devices, because when a removable memory mappable device is disconnected from a computing device, the virtual address space associated with the removable memory mappable device may be released after the physical address space associated with the removable memory mappable device is released. Specifically, as described on page 3, lines 14-33 of Appellant's specification, as originally filed, when a removable memory mappable device is disconnected from a computing device, a memory management system of an operating system of the computing device releases the physical address space associated with the removable memory mappable device according to the semantics of the operating system. However, a process associated with the removable memory mappable device may not have yet released the virtual address space associated with that physical address space at the time of disconnection.

As such, Appellant respectfully submits that each and every element and limitation of independent claims 1, 8, 19, and 22-23 is not taught, suggested, or made obvious in view of, the combination of the Arimilli '494 and Browning '023 references. Accordingly, Appellant respectfully requests reconsideration and

withdrawal of the § 103 rejection of independent claims 1, 8, 19, and 22-23, as well as those claims that depend therefrom.

2. Arguments regarding independent claim 13.

a. For independent claim 13, the Arimilli '494 reference does not teach, suggest, or render obvious each and every element and limitation.

With regard to independent claim 13, the Examiner states on page 8 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses "means for un-mapping a virtual address space, i.e., processor's move engine (28, figure 3)." However, the Abstract of the Arimilli '494 reference states:

> A processor contains a move engine and a memory controller contains a mapping engine that, together, transparently reconfigure physical memory to accomplish addition, subtraction, or replacement of a memory module.

Column 11, lines 6-15 of the Arimilli '494 reference goes on to state:

> As will be appreciated, the preferred embodiment provides for a memory module to be inserted, removed or replaced in physical memory 22 without the operating system having to direct and control the reconfiguration of physical memory to accomplish the physical memory change. In the preferred embodiment, move engine 28 and mapping engines 26, 36, 46 work in conjunction to transparently reconfigure the physical memory to accomplish the addition, subtraction, or replacement of a particular memory module in the physical memory.

By so stating, the Arimilli '494 reference appears to teach that the move engine and mapping engines transparently reconfigure physical memory to

accomplish a physical memory change. Hence, the Arimilli '494 reference does not teach, "means for unmapping a virtual address space," as recited in Appellant's independent claim 13.

b. For independent claim 13, the Browning '023 reference does not teach, suggest, or render obvious each and every element and limitation.

The Examiner also states on page 9 of the June 19, 2008, Final Office Action:

> Arimilli differs from the claimed invention in not specifically teaching that means for un-mapping the virtual address space for the process that is triggered as a physical address space used by the process is being released.

However, the Examiner cites the Browning '023 reference as teaching the above quoted language on page 9 of the Final Office Action.

The Browning '023 reference appears to describe, "invalidating specified pretranslations maintained in a data processing system which maintains decentralized copies of pretranslations." (Abstract). Column 1, lines 30-36, of the Browning '023 reference appears to describe pretranslations as:

> To translate is the process of looking up from a centralized record the physical address to which a particular virtual address is mapped. Thus, a pretranslation is obtained using the centralized record. A pretranslation is a copy of the translation. The pretranslation may be stored, such as with a virtual buffer, for later use in order to avoid the process of translation.

By so stating, the Browning '023 reference appears to teach storing a number of pretranslations, such as with a virtual buffer, for later use in order to avoid the process of translation, which also avoids accessing the virtual address space. Hence, the Browning '023 reference does not teach, "means for unmapping a virtual address space," as recited in Appellant's independent claim 13.

Additionally, column 8, line 45, through column 9, line 10, of the Browning '023 reference appears to teach a process in which a kernel receives an acknowledgment of an interprocessor interrupt and invalidates the relevant pretranslation lists before memory migration and removal, e.g., release, of the real, e.g., physical, pages of memory, as discussed above with regard to independent claims 1, 8, 19, and 22-23.

In contrast, Appellant's independent claim 13 recites:

> means for unmapping a virtual address space for a process that is triggered as a physical address space used by the process is being released, in a manner which does not violate semantics for an operating system of the computing device, when a removable memory mappable device associated with the process is logically disconnected.

Further, Appellant respectfully submits that independent claim 13 addresses, at least in part, memory management for removable memory mappable devices, e.g., devices that can be disconnected from a computing device. As discussed above with regard to independent claims 1, 8, 19, and 22-23, the Browning '023 reference does not appear to address removable memory mappable devices, and the process taught in column 8, line 45, through column 9, line 10, of the Browning '023 reference is not usable in the context of removable memory mappable devices.

As such, Appellant respectfully submits that each and every element and limitation of independent claim 13 is not taught, suggested, or made obvious in view of the combination of the Arimilli '494 and Browning '023 references. Accordingly, Appellant respectfully requests reconsideration and withdrawal of the § 103 rejection of independent claim 13, as well as those claims that depend therefrom.

3. Arguments regarding dependent claims 3-5 and 7.

a. For dependent claims 3-5 and 7, the cited references do not teach, suggest, or render obvious each and every element and limitation.

Claims 3-5 and 7 depend from independent claim 1. As presented above, Appellant respectfully submits that independent claim 1 is in condition for allowance. As such, Appellant respectfully submits that dependent claims 3-5 and 7 are also allowable. Moreover, as presented below, Appellant respectfully submits that the Arimilli '494 and Browning '023 references do not teach, suggest, or render obvious each and every element and limitation of dependent claims 3-5 and 7.

With regard to dependent claim 3, the Examiner states on page 5 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that "the virtual address space includes an input/output space (col. 7 lines 58-65)." However, column 7, lines 58-65 of the Arimilli '494 reference states:

> Each of mapping engines 26, 36, and 46 contain registers 301, 305, 309, respectively, storing a "current" real address for its associated memory module and a "new" real address for its associated memory module (as used herein, the real address refers to

> the entire real address or that portion (for example, the higher-order bits) needed to uniquely identify an associated memory module storing data addressed by the indexed block of memory).

By so stating, the Arimilli '494 reference appears to teach that each mapping engine contains registers which store both a "current" real address and a "new" real address for the memory module associated with each mapping engine. Hence, the Arimilli '494 reference does not teach that the virtual address space includes an input/output space.

In contrast, Appellant's dependent claim 3 recites, "The computing device of claim 1, wherein the virtual address space includes an input/output space." As noted above, from Appellant's review of the Browning '023 reference, the Browning '023 reference does not teach that "the virtual address space includes an input output space," as recited in Appellant's dependent claim 3.

With regard to dependent claim 4, the Examiner states on page 5 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that:

> the program instructions are part of a memory management system, which includes a virtual memory data structure associated with the process (col. 6 line 66 through col. 7 line 15).

However, column 6, line 66, through column 7, line 15, of the Arimilli '494 reference states:

> In accordance with a preferred embodiment, physical mapping 212 is then performed by memory controllers 24, 34 and 44. Physical mapping 212 translates the real addresses for the address pages P1-RA, P2-RA, and P3-RA and maps them into the corresponding physical addresses, P1-PA, P2-PA, and P3-PA,

> respectively, representing the physical addresses of those requested pages within the corresponding memory modules M1, M2 and M3. The physical address indicates the specific memory location within the memory module storing the addressed information. For example, P2-PA specifies the specific row and column addresses to uniquely identify the addressed page in memory module 2. This physical mapping mechanism is invisible from the operating system OS, which views all of the physical memory resources by means of their real addresses without a priori distinguishing the locality of these resources to a particular memory module M1, M2 and M3 in system memory 22.

By so stating, the Arimilli '494 reference appears to teach a physical mapping process in which the real addresses for the address pages are mapped into physical addresses. Hence, the Arimilli '494 reference does not teach a memory management system which includes a virtual memory data structure associated with the process.

In contrast, Appellant's dependent claim 4 recites, "The computing device of claim 1, wherein the program instructions are part of a memory management system which includes a virtual memory data structure associated with the process." As noted above, from Appellant's review of the Browning '023 reference, the Browning '023 reference does not teach "a memory management system which includes a virtual memory data structure associated with the process," as recited in Appellant's dependent claim 4.

With regard to dependent claim 5, the Examiner states on page 5 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that:

> the program instructions execute to register the virtual address space is no longer valid for process use in the virtual memory data structure associated with the process (col. 8 lines 9-26).

However, column 8, lines 9-26 of the Arimilli '494 reference states:

> Move engine 28 loads each register 301, 305, 309 as necessary to perform the memory re-configuration. Field 302 shows that memory module M1's current real address is RA1. Field 304 contains the new real address for memory module M1, and shows that it remains the same at RA1. Mapping engine 36 contains field 306, showing the current real address of memory module M1 as RA2. For its new real address, memory module M2 is given a real address that is outside the total real address space currently allocated to the physical memory system. Thus, for example, field 308 contains a new real address for memory module M2, that is RA4, which is outside the current real address space (e.g. 0-128 GB) as mapped to the real addresses (i.e. RA1-RA2). Similarly, move engine 28 has assigned memory module M3 the previous real address of memory module M2, as shown in field 312, indicating memory module M3's new real address as RA2. Field 310 shows memory module M3's current real address of RA3, which is now outside the addressable real space for the operating system.

By so stating, the Arimilli '494 reference appears to teach that during a memory re-configuration, the new real address for a memory module may remain the same as its current real address, or it may be a different real address that is outside the total real address space currently allocated to the physical memory system. Hence, the Arimilli '494 reference does not teach that the program instructions execute to register the virtual address space is no longer valid for

process use in the virtual memory data structure associated with the process. Indeed, as previously discussed with regard to independent claims 1, 8, 19, and 22-23, the Examiner admits that the Arimilli '494 reference does not disclose "to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use."

In contrast, Appellant's dependent claim 5 recites:

> The computing device of claim 4, wherein the program instructions execute to register the virtual address space is no longer valid for process use in the virtual memory data structure associated with the process.

As noted above, from Appellant's review of the Browning '023 reference, the Browning '023 reference does not describe "to register the virtual address space is no longer valid for process use in the virtual memory data structure associated with the process," as recited in Appellant's dependent claim 5.

With regard to dependent claim 7, the Examiner states on page 5 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that, "the program instructions execute to register that the virtual address space is available for use when the process releases the virtual address space (col. 7 lines 32-57)." However, column 7, lines 32-57 of the Arimilli '494 reference states:

> FIG. 3 illustrates an embodiment where a memory module is being removed from physical memory in a simplified drawing of data processing system 8. As will be explained, the processor's move engine works in conjunction with the associated mapping engines to take the associated memory module off-line prior to its physical removal. Generally, the move engine copies the contents of

> the memory module to be removed into the remaining memory modules of physical memory. Then, the real address of the old memory module is re-assigned to the new memory module receiving the copied contents.
>
> In this example, memory module M2 is being removed from data processing system 8. As a first step, processor unit 10 reports to the operating system that its total available physical memory has now been reduced by one memory module. For example, if each memory module M1, M2, M3 is a 64 Giga-Byte (GB) memory device, the operating system would be informed that its available physical memory is now 128 GB. The operating system immediately begins to swap out pages to reduce the amount of stored data accordingly. Processor unit 10 notifies move engine 28 and mapping engines 26, 36, 46 that memory module M2 is being removed from physical memory 22. Move engine 28 immediately selects the remaining module or modules that will be used to store the data contained in memory module M2.

By so stating, the Arimilli '494 reference appears to teach that when a memory module is removed from the physical memory, the memory module's contents are copied into the remaining memory modules, the real address of the memory module is re-assigned to the new memory module receiving the copied contents, and the operating system is informed that its total available physical memory has been reduced. Hence, the Arimilli '494 reference does not teach that the program instructions execute to register that the virtual address space is available for use when the process releases the virtual address space.

In contrast, Appellant's dependent claim 7 recites:

> The computing device of claim 1, wherein the program instructions execute to register that the virtual address space is available for use when the process releases the virtual address space.

As noted above, from Appellant's review of the Browning '023 reference, the Browning '023 reference does not teach "to register that the virtual address space is available for use when the process releases the virtual address space," as recited in Appellant's dependent claim 7.

As such, Appellant respectfully submits that each and every element and limitation of dependent claims 3-5 and 7 is not taught, suggested, or made obvious in view of the combination of the Arimilli '494 and Browning '023 references. Accordingly, Appellant respectfully requests reconsideration and withdrawal of the § 103 rejection of dependent claims 3-5 and 7.

4. Arguments regarding dependent claims 9 and 11-12.

a. For dependent claims 9 and 11-12, the cited references do not teach, suggest, or render obvious each and every element and limitation.

Claims 9 and 11-12 depend from independent claim 8. As presented above, Appellant respectfully submits that independent claim 8 is in condition for allowance. As such, Appellant respectfully submits that dependent claims 9 and 11-12 are also allowable. Moreover, as presented below, Appellant respectfully submits that the Arimilli '494 and Browning '023 references do not teach, suggest, or render obvious each and every element and limitation of dependent claims 9 and 11-12.

With regard to dependent claim 9, the Examiner states on page 7 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that:

> the program instructions execute to un-map the virtual address space in a manner which do not violate semantics for an operating system of the computing device (abstract and col. 11 lines 6-26).

However, as discussed above with regard to independent claim 13, the Arimilli '494 reference does not teach means for unmapping a virtual address space.

In contrast, Appellant's dependent claim 9 recites:

> The computing device of claim 8, wherein the program instructions execute to unmap the virtual address space in a manner which do not violate semantics for an operating system of the computing device.

As noted above with regard to independent claim 13, the Browning '023 reference does not teach means for unmapping a virtual address space.

With regard to dependent claim 11, the Examiner states on page 8 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that:

> the program instructions execute to allow the process to un-map the virtual address space subsequent to the release of the physical address space... (col. 7 lines 17-42).

However, as discussed above with regard to independent claim 13, the Arimilli '494 reference does not teach means for unmapping a virtual address space.

In contrast, Appellant's dependent claim 11 recites:

> The computing device of claim 8, wherein the program instructions execute to allow the process to unmap the virtual

> address space subsequent to the release of the physical address space.

As noted above with regard to independent claim 13, the Browning '023 reference does not teach means for unmapping a virtual address space.

With regard to dependent claim 12, the Examiner states on page 8 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that:

> the program instruction execute to... indicate an operation as failed if the process attempts to perform the operation subsequent to registering that the virtual address space is no longer valid for process use (col. 7 lines 17-42).

However, column 7, lines 17-42 of the Arimilli '494 reference states:

> Referring back to FIG. 3, move engines 28 and mapping engines 26, 36 and 46 provide the virtualization function of the physical memory to allow efficient re-configuration of the physical memory 22, in accordance with the preferred embodiment. When physical memory 22 is re-configured, such as when one of memory modules M1, M2 and M3 are inserted, removed or replaced in the system, move engine 28 performs a data transfer between the memory modules of physical memory 22. Mapping engines 26, 36, 46 control the real-to-physical addressing of memory modules M1, M2, M3 to allow the addition, subtraction or substitution of a particular memory module. This memory management is done efficiently at the hardware/firmware level, requiring little operating system resources to accomplish the re-configuration of physical memory.
>
> FIG. 3 illustrates an embodiment where a memory module is being removed from physical memory in a simplified drawing of data processing system 8. As will be explained, the processor's

> move engine works in conjunction with the associated mapping engines to take the associated memory module off-line prior to its physical removal. Generally, the move engine copies the contents of the memory module to be removed into the remaining memory modules of physical memory. Then, the real address of the old memory module is re-assigned to the new memory module receiving the copied contents.

By so stating, the Arimilli '494 reference appears to teach that during a re-configuration of the physical memory, the move engine performs a data transfer between the memory modules of the physical memory by copying the contents of the memory module to be removed into the remaining memory modules, and the mapping engines control the real-to-physical addressing of the memory modules by re-assigning the real address of the old memory module to the new memory module receiving the copied contents. Hence, the Arimilli '494 reference does not teach indicating an operation as failed if the process attempts to perform the operation subsequent to either registering that the virtual address space is no longer valid for process use, or the device being logically disconnected from the computing device.

In contrast, Appellant's dependent claim 12 recites:

> The computing device of claim 8, wherein the program instructions execute to indicate an operation as failed if the process attempts to perform the operation subsequent to registering that the virtual address space is no longer valid for process use.

As noted above, from Appellant's review of the Browning '023 reference, the Browning '023 reference does not describe indicating an operation as failed if the process attempts to perform the operation subsequent to either registering that the

virtual address space is no longer valid for process use, or the device being logically disconnected from the computing device, as recited in Appellant's dependent claim 12.

As such, Appellant respectfully submits that each and every element and limitation of dependent claims 9 and 11-12 is not taught, suggested, or made obvious in view of the combination of the Arimilli '494 and Browning '023 references. Accordingly, Appellant respectfully requests reconsideration and withdrawal of the § 103 rejection of dependent claims 9 and 11-12, as well as those claims that depend therefrom.

5. Arguments regarding dependent claims 16 and 18.

a. For dependent claims 16 and 18, the cited references do not teach, suggest, or render obvious each and every element and limitation.

Claims 16 and 18 depend from independent claim 13. As presented above, Appellant respectfully submits that independent claim 13 is in condition for allowance. As such, Appellant respectfully submits that dependent claims 16 and 18 are also allowable. Moreover, as presented below, Appellant respectfully submits that the Arimilli '494 and Browning '023 references do not teach, suggest, or render obvious each and every element and limitation of dependent claims 16 and 18.

With regard to dependent claim 16, the Examiner states on page 10 of the June 19, 2008, Final Office Action that the Browning '023 reference discloses:

> the means for un-mapping the virtual address space includes program instructions which execute to remove a mapping of the object to physical memory (col. 8 line 45 through col. 9 line 27).

As discussed above with regard to independent claims 1, 8, 19, and 22-23, the Browning '023 reference appears to teach a process in which a kernel first sends and receives acknowledgment of an interprocessor interrupt to and from all CPUs, then scans all registered RPN lists and invalidates all entries that correspond to real pages that are within the range of memory to be removed, and finally performs memory migration and removal of real pages of memory. That is, the Browning '023 reference appears to teach that real pages of memory are removed. Hence, the Browning '023 reference does not teach that a mapping of the object to physical memory is removed.

In contrast, Appellant's dependent claim 16 recites:

> The computing device of claim 15, wherein the means for unmapping the virtual address space includes program instructions which execute to remove a mapping of the object to physical memory.

As noted above, from Appellant's review of the Arimilli '494 reference, the Arimilli '494 reference does not teach "to remove a mapping of the object to physical memory," as recited in Appellant's dependent claim 16.

With regard to Appellant's dependent claim 18, the Examiner states on page 10 of the June 19, 2008, Final Office Action that the Browning '023 reference discloses that "the program instructions execute to set a bit in the region [sic] of the

virtual memory data structure to indicate that the virtual address space is not available for use (col. 8 line 45 through col. 9 line 27)."

As discussed above with regard to independent claims 1, 8, 19, and 22-23, the Browning '023 reference appears to teach a process in which a kernel first sends and receives acknowledgment of an interprocessor interrupt to and from all CPUs, then scans all registered RPN lists and invalidates all entries that correspond to real pages that are within the range of memory to be removed, and finally performs memory migration and removal of real pages of memory. Hence, the Browning '023 reference does not teach to set a bit in a pregion of the virtual memory data structure to indicate that the virtual address space is not available for use.

In contrast, Appellant's dependent claim 18 recites:

> The computing device of claim 17, wherein the program instructions execute to set a bit in a pregion of the virtual memory data structure to indicate that the virtual address space is not available for use.

As noted above, from Appellant's review of the Arimilli '494 reference, the Arimilli '494 reference does not teach "to set a bit in a pregion of the virtual memory data structure," as recited in Appellant's dependent claim 18.

As such, Appellant respectfully submits that each and every element and limitation of dependent claims 16 and 18 is not taught, suggested, or made obvious in view of the combination of the Arimilli '494 and Browning '023 references. Accordingly, Appellant respectfully requests reconsideration and withdrawal of the § 103 rejection of dependent claims 16 and 18, as well as those claims that depend therefrom.

6. Arguments regarding dependent claims 20-21.

a. For dependent claims 20-21, the cited references do not teach, suggest, or render obvious each and every element and limitation.

Claims 20-21 depend from independent claim 19. As presented above, Appellant respectfully submits that independent claim 19 is in condition for allowance. As such, Appellant respectfully submits that dependent claims 20-21 are also allowable. Moreover, as presented below, Appellant respectfully submits that the Arimilli '494 and Browning '023 references do not teach, suggest, or render obvious each and every element and limitation of dependent claims 20-21.

With regard to dependent claim 20, the Examiner states on pages 8 and 10 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that:

> the program instructions execute to allow the process to un-map the virtual address space subsequent to the release of the physical address space... (col. 7 lines 17-42).

However, as discussed above with regard to dependent claim 11, the Arimilli '494 reference does not teach means for unmapping a virtual address space.

In contrast, Appellant's dependent claim 20 recites;

> The method of claim 19, further including unmapping the virtual address space at the request of the process subsequent to the device being logically disconnected from the computing device.

As noted above with regard to dependent claim 11, the Browning '023 reference does not teach means for unmapping a virtual address space.

With regard to dependent claim 21, the Examiner states on pages 8 and 10 of the June 19, 2008, Final Office Action that the Arimilli '494 reference discloses that:

> the program instruction execute to... indicate an operation as failed if the process attempts to perform the operation subsequent to registering that the virtual address space is no longer valid for process use (col. 7 lines 17-42).

However, as discussed above with regard to dependent claim 12, the Arimilli '494 reference does not teach indicating an operation as failed if the process attempts to perform the operation subsequent to either registering that the virtual address space is no longer valid for process use, or the device being logically disconnected from the computing device.

In contrast, Appellant's dependent claim 21 recites:

> The method of claim 19, further including indicating an operation as failed if the process attempts to perform the operation subsequent the device being logically disconnected from the computing device.

As noted above with regard to dependent claim 12, the Browning '023 reference does not teach indicating an operation as failed if the process attempts to perform the operation subsequent to either registering that the virtual address space is no longer valid for process use, or the device being logically disconnected from the computing device, as recited in Appellant's dependent claim 21.

As such, Appellant respectfully submits that each and every element and limitation of dependent claims 20-21 is not taught, suggested, or made obvious in

view of the combination of the Arimilli '494 and Browning '023 references. Accordingly, Appellant respectfully requests reconsideration and withdrawal of the § 103 rejection of dependent claims 20-21.

CONCLUSION

Appellant respectfully submits that the claims are in condition for allowance and notification to that effect is earnestly requested. The Examiner and/or members of the Board are invited to telephone Appellant's attorney Edward J. Brooks III at (612) 236-0120 to facilitate this appeal.

At any time during the pendency of this application, please charge any additional fees or credit overpayment to the Deposit Account No. 08-2025.

CERTIFICATE UNDER 37 C.F.R. §1.8: The undersigned hereby certifies that this correspondence is being transmitted to the United States Patent and Trademark Office facsimile number (571) 273-8300. on this 30 day of July, 2008.

Name: Jennifer L. Vonhof

Signature:

Respectfully Submitted,
Manish Ahluwalia

By his Representatives:

Brooks, Cameron & Huebsch, PLLC
1221 Nicollet Avenue, Suite 500
Minneapolis, MN 55403

Atty: Edward J. Brooks III
Reg. No.: 40,925

Date: 7/30/2008

VIII. CLAIMS APPENDIX

1. (Previously Presented) A computing device, comprising:

a processor;

a memory coupled to the processor; and

program instructions provided to the memory and executable by the processor to:

track a virtual address space for a process associated with a device connected to the computing device;

release a physical address space associated with the virtual address space when the device has a connection removed from the computing device;

register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use;

wherein registering is triggered by detection that the physical address space that was being used by processes associated with the device has been released; and

wherein the registering occurs as the physical address space is released and before release of the virtual address space by the process.

2. (Original) The computing device of claim 1, wherein the device includes a device which can be mapped to memory.

3. (Original) The computing device of claim 1, wherein the virtual address space includes an input/output space.

4. (Original) The computing device of claim 1, wherein the program instructions are part of a memory management system which includes a virtual memory data structure associated with the process.

5. (Original) The computing device of claim 4, wherein the program instructions execute to register the virtual address space is no longer valid for process use in the virtual memory data structure associated with the process.

6. (Original) The computing device of claim 1, wherein the program instructions execute to allocate the virtual address space when the process requests physical memory.

7. (Original) The computing device of claim 1, wherein the program instructions execute to register that the virtual address space is available for use when the process releases the virtual address space.

8. (Previously Presented) A computing device, comprising:

a processor;

a random access memory coupled to the processor; and

program instructions provided to the memory and executable by the processor, the program instructions are part of a memory management system to:

dereference a virtual address space for a process associated with a removable memory mappable device connected to the computing device;

release a physical address space associated with the virtual address space when the device associated with the process is logically disconnected; and

register by providing an indication in the virtual memory data structure for the process that the virtual address space is no longer available to the process;

wherein to register is triggered by detection that the physical address space that was being used by processes associated with the device has been released; and

wherein to register occurs as the physical address space is released and before release of the virtual address space by the process.

9. (Previously Presented) The computing device of claim 8, wherein the program instructions execute to unmap the virtual address space in a manner which do not violate semantics for an operating system of the computing device.

10. (Original) The computing device of claim 9, wherein the operating system is selected from the group of a Unix operating system and a Linux operating system.

11. (Original) The computing device of claim 8, wherein the program instructions execute to allow the process to unmap the virtual address space subsequent to the release of the physical address space.

12. (Original) The computing device of claim 8, wherein the program instructions execute to indicate an operation as failed if the process attempts to perform the operation subsequent to registering that the virtual address space is no longer valid for process use.

13. (Previously Presented) A computing device, comprising:

a processor;

a memory coupled to the processor, the memory including program instructions for maintaining a virtual memory data structure as part of a memory management system; and

means for unmapping a virtual address space for a process that is triggered as a physical address space used by the process is being released, in a manner which does not violate semantics for an operating system of the computing device, when a removable memory mappable device associated with the process is logically disconnected.

14. (Original) The computing device of claim 13, wherein the program instructions execute to dereference the virtual address space for the process.

15. (Previously Presented) The computing device of claim 13, wherein the means for unmapping the virtual address space includes program instructions which execute to maintain a representation of an object associated with the process in the virtual memory data structure of the process.

16. (Previously Presented) The computing device of claim 15, wherein the means for unmapping the virtual address space includes program instructions which execute to remove a mapping of the object to physical memory.

17. (Previously Presented) The computing device of claim 16, wherein the means for unmapping the virtual address space includes program instructions which execute to register in the virtual memory data structure of the process that the virtual address space associated with the process is not available for use subsequent to when the mapping of the object to physical memory has been removed.

18. (Previously Presented) The computing device of claim 17, wherein the program instructions execute to set a bit in a pregion of the virtual memory data structure to indicate that the virtual address space is not available for use.

19. (Previously Presented) A method for memory management on a computing device, comprising:

dereferencing a memory address for a process associated with a removable memory mappable device;

mapping a representation of an object associated with the process in a virtual memory data structure associated with the process;

removing the object from physical memory when the device is logically disconnected from the computing device; and

providing an indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process as triggered by detection of a physical address space used by the process being released and

when the object is removed from physical memory, without removing the representation of the object from the virtual memory data structure for the process.

20. (Original) The method of claim 19, further including unmapping the virtual address space at the request of the process subsequent to the device being logically disconnected from the computing device.

21. (Original) The method of claim 19, further including indicating an operation as failed if the process attempts to perform the operation subsequent the device being logically disconnected from the computing device.

22. (Previously Presented) A method for memory management, comprising:

tracking a virtual address space for a process associated with a removable memory mappable device connected to a computing device;

releasing a physical address space when the device has a logical connection removed from the computing device; and

at the release of the physical address space used by the process and before the process has released the virtual address space, registering an indication in a virtual memory data structure for the process that the virtual address space is not available to the process in a manner which does not violate semantics of an operating system.

23. (Previously Presented) A computer readable medium having computer readable instructions stored thereon for execution by a device to perform a method, comprising:

dereferencing a virtual address space for a process associated with a removable memory mappable device as part of a memory management system on a computing device;

releasing a physical address space when the device is logically disconnected from the computing device; and

at the release of the physical address space used by the process and before the process has released the virtual address space, registering an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process in a manner which does not violate semantics for an operating system the computing device.

IX. EVIDENCE APPENDIX

None

X. RELATED PROCEEDINGS APPENDIX

None